01-T2 Minecraft Game Mode

CS 4850 | Section 03
Spring 2025 | March 20th
Professor Sharon Perry

Ashley Ahn
Matthew Elledge
AnnaGrace Gwee
Bryan Nguyen
Logan Slicker

Preface

The KSU Minecraft server hosts many game modes and is looking for more: Thus, our game mode: **Minecraft Tower Defense!**

Our goal is to introduce a game mode akin to the Tower Defense Genre, inspired by elements of BloonsTD6 and Sanctum 2, alongside some homebrew twists to it!

Overview

- **Requirements**
  - Functional & Non-Functional
- **Design**
  - Tools & Constraints
  - Architecture
- **Development**
  - Gameplay & Mechanics
  - Database Connections
  - Map Designs
- **Demo**
  - Manual Compilation
  - Current Prototype
  - Structure
- **Outlook**
  - What's Next?

Requirements

Requirements

Functional

- Map Selection
- Round Management
- Mob Management
- Tower Management
- Player Management

Non-Functional

- Capacity
  - Light Performance
- Usability
  - Ease of Implementation
- Modifiability
  - Configurable & Updatable

Design

Design – Tools & Constraints

Tools

- Plugins
  - SimpleScore, Parties, WorldEdit, PlaceholderAPI
- APIs
  - Paper, Parties, PlaceholderAPI
- IntelliJ MC Dev Plugin
  - Basis for MC Plugins

Constraints

- Asynchronous
- Configurable
  - Config File
  - Maintainable
  - Expandable
- Performance
  - Server Limited

Design – Architecture

- **Our Game Mode will be coded in Java**
  - Used by Minecraft and Paper, the server type the KSU MC Server is hosted on & popular framework for MC Plugins
- **Utilizing the Paper API Framework**
  - Allows us to easily create GUIs, utilize built-in functions, and use their event, adventure, and entity APIs to modify game behavior to fit our game mode
- **System Design**
  - Proper command structure for debugging & server modification
  - Effective memory management to ensure server performance

Development

Development – Gameplay & Mechanics

Gameplay

- Players have a limited number of hearts
- Prevent mobs from reaching the end path
- Survive waves of mobs (Count depending on map)
- Utilize tools & towers

Mechanics

- Players can use weapons to fight incoming mobs
- Players can spend coins to build towers to combat mobs
- Mobs are to follow the path
  - Defeated mobs provide Coins
  - Mobs reaching the end reduce the player's hearts

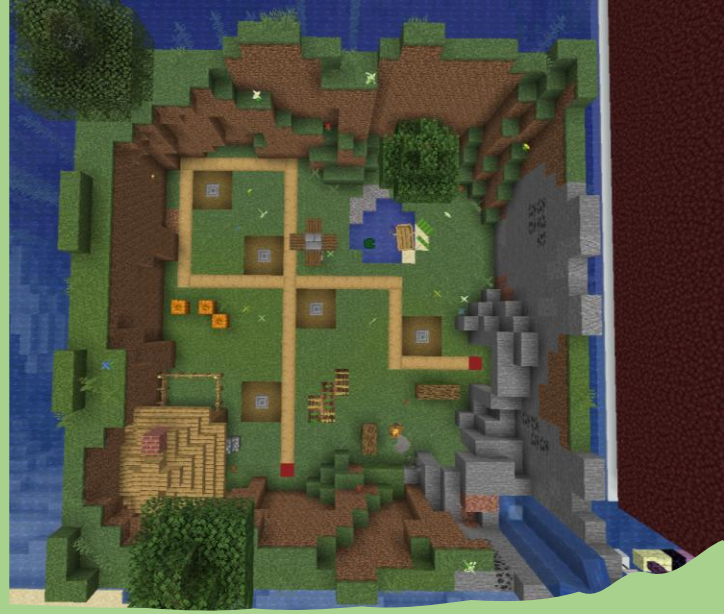
Development – Database Connections

Database of choice: SQLite

- SQLite is very lightweight, free, and relatively easy to use and implement

Database applications:

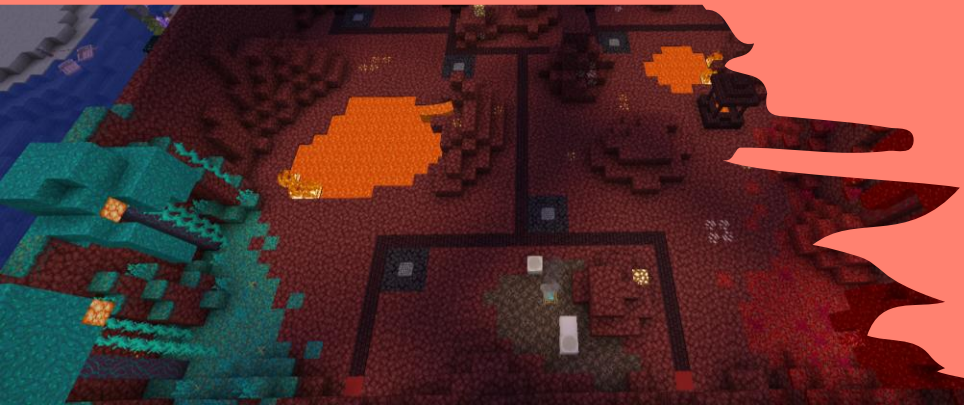
- Collection of game data for user concurrency and data tracking
- Will be used to create the plugin's leaderboards system
  - Tracks player's fastest win time on each map, total wins, coins gained, coins spent, and towers/upgrades purchased



Map 1 - Tutorial

Map 2
Cherry Blossom Garden

Map 3 - Cityscape

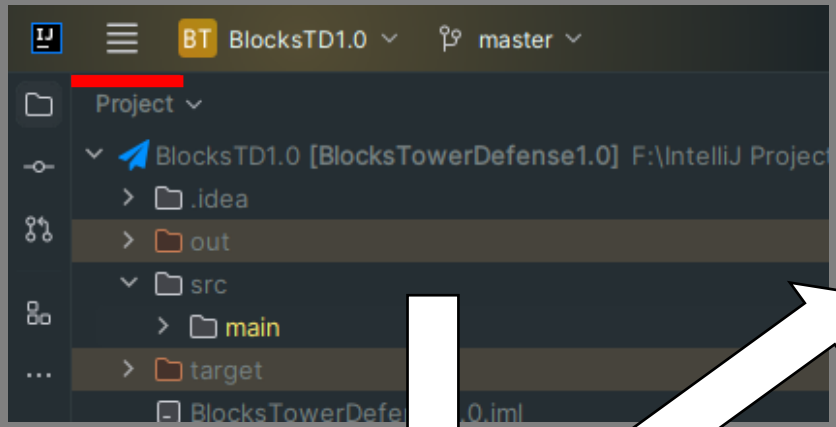


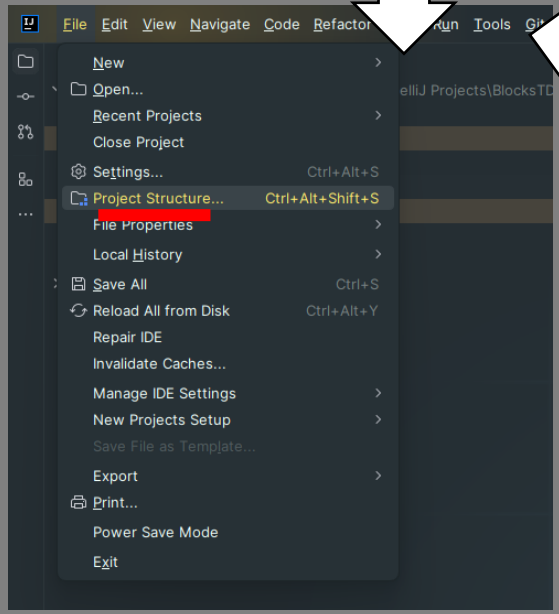
Map 4 - Nether

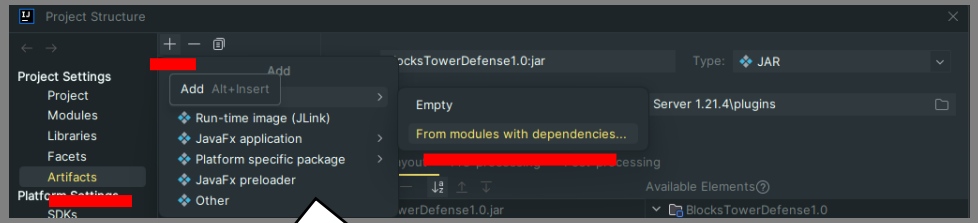
Demo

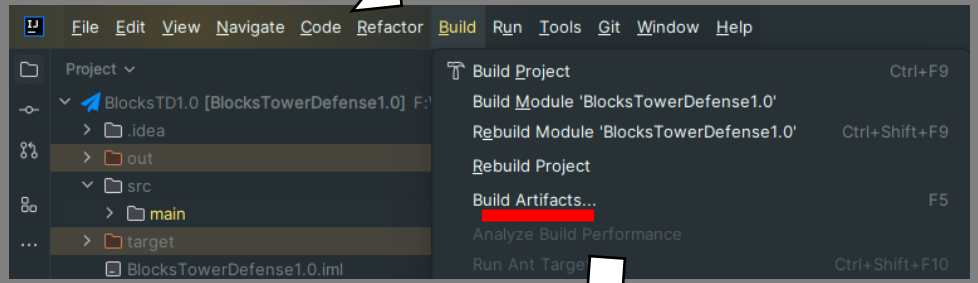
Demo – Manual Compilation

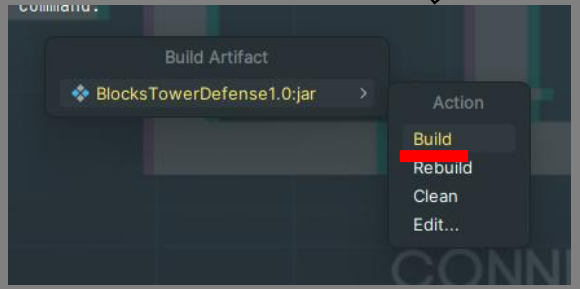
- **Step 1: Compile the plugin (IntelliJ)**
  - Open the project in IntelliJ, from the top left menu, select "File", "Project Structure", then under "Project Settings", select "Artifacts"
  - Create a new artifact with the plus sign (+) at the top-left of the window, selecting it as a JAR file, with "From modules with dependencies", and click OK at the pop-up
    - ❑ Have the output directory of the artifact build be the server's plugin folder
- **Step 2: Build the plugin into the server**
  - Once the build location is correct, go to the menu at the top-left of the screen, and select "Build", then "Build Artifacts".
  - On the pop-up, it will show the artifact created. Select "Build" and wait.
- **Step 3: Run the server & Verify**
  - To ensure the plugin is active, type "/plugins", in which it should display the plugin.


BlocksTD1.0
master
Project
BlocksTD1.0 [BlocksTowerDefense1.0] F:\IntelliJ Project
.idea
out
src
main
target


File Edit View Navigate Code Refactor
New
Open...
Recent Projects
Close Project
Settings... Ctrl+Alt+S
Project Structure... Ctrl+Alt+Shift+S
File Properties
Local History
Save All Ctrl+S
Reload All from Disk Ctrl+Alt+Y
Repair IDE
Invalidate Caches...
Manage IDE Settings
New Projects Setup
Save File as Template...
Export
Print...
Power Save Mode
Exit


Project Structure
Project Settings
Project
Modules
Libraries
Facets
Artifacts
SDKs
Add Alt+Insert
Run-time image (JLink)
JavaFx application
Platform specific package
JavaFx preloader
Other
Type: JAR
Empty
From modules with dependencies...
Server 1.21.4\plugins
Available Elements


File Edit View Navigate Code Refactor Build Run Tools Git Window Help
Project
BlocksTD1.0 [BlocksTowerDefense1.0]
.idea
out
src
main
target
BlocksTowerDefense1.0.iml
Build Project Ctrl+F9
Build Module 'BlocksTowerDefense1.0'
Rebuild Module 'BlocksTowerDefense1.0' Ctrl+Shift+F9
Rebuild Project
Build Artifacts... F5
Analyze Build Performance
Run Ant Target Ctrl+Shift+F10


Build Artifact
BlocksTowerDefense1.0:jar
Action
Build
Rebuild
Clean
Edit...

Current Prototype

Demo – Structure

- Main
  - Commands
    - Handles all of our command logic for operations like /mtd party, and /mtd hub
  - Logic
    - Game logic
      - Almost everything here is dealing with async calls: It is what runs when you do /startgame <mapname>, like our MobHandler, SummonTower, and PlayerEventHandlers
    - Static logic
      - General logic that we call from game logic that can be handled synchronously, such as the Economy, teleportation, and database classes
  - Maps
    - Class dedicated to modifying the maps.yml config. This tells the newly spawned mobs what map they are on and what waypoints to path towards
  - Main
    - Initializes all of our commands and events that are required for the game to function

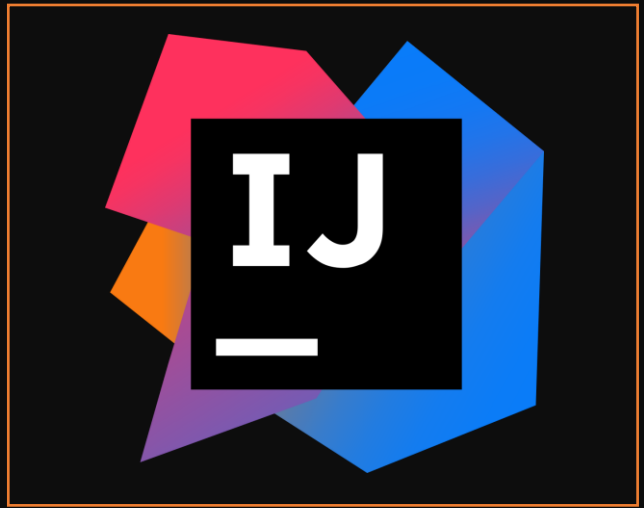
Outlook

Outlook – What's Next?

- Functionality
  - Bug Testing & Code Cleanup
  - Singleplayer & Multiplayer
  - Config for All Game Options
- Additions
  - More Expansive Hub Zone
  - Tower Animations
  - Leaderboards System
  - Extra Map Functionality
    - ❑ All Maps up to Level 100
  - KSU Server Playtest Feedback
- Mobs
  - Balancing Difficulty & Economy
  - Adding Mob Variety
- Towers
  - Tower Variety & Identity
  - Branching Upgrade Paths
- Players
  - Weapon Upgrade Paths
  - Weapon Enchantments
  - Consumable Items

Conclusion

Through the usage of IntelliJ as our IDE alongside our plugins and APIs (such as Paper), we hope to properly implement a fun and functional Tower Defense game mode for the KSU Minecraft server that will have longevity with its future balance changes and expansions!



Discord Link to the KSU Minecraft Server:
https://discord.gg/7xWVtaDvzK

We submitted our power point slides prior to this presentation

Questions and Feedback?